The Case of Performance Variability on Dragonfly-based Systems

Presented by Carlos Venegas

The Problem

- Lack of predictability in run times
- Harder to optimize code
- Wide range of causes

Proposed Solution

- Analysis of the relevance of factors to variability on Dragonfly
- Create a way to predict variability
- Why Dragonfly?

Context

- Focus on Cray XC

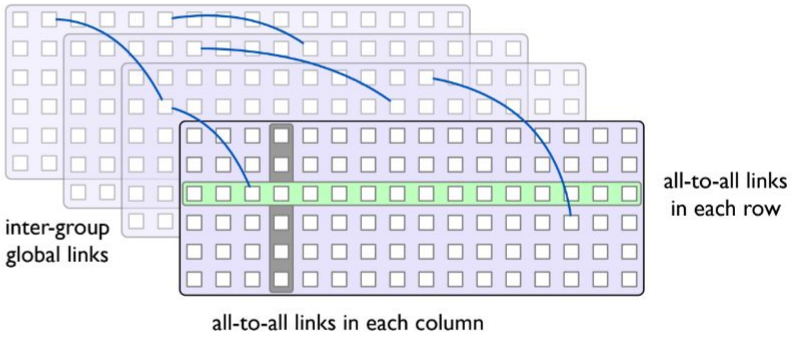

Fig. 2. The dragonfly network configuration in Cray XC systems.

Data

- Ran 4 programs:
  - AMG
  - MILC
  - miniVite
  - UMT
- Considered Data:
  - Performance stats
  - Location

Computation vs. Communication

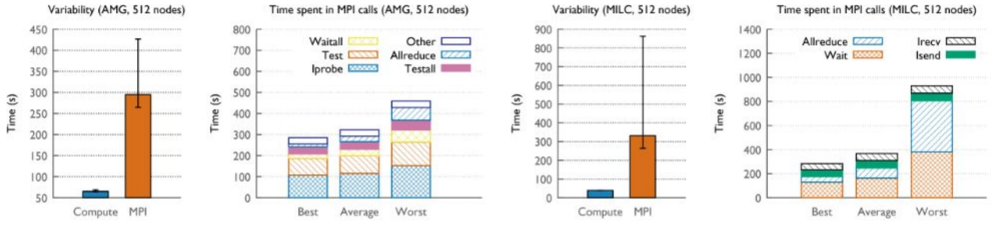

Fig. 4. Time spent in computation and communication, and in different MPI routines in AMG (left plots) and MILC (right plots) on 512 nodes.

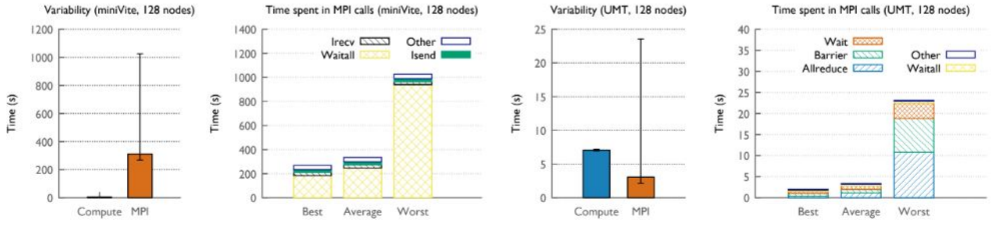

Fig. 5. Time spent in computation and communication, and in different MPI routines in miniVite (left plots) and UMT (right plots) on 128 nodes

Applying the Data

- Measured:
  - Other running jobs
  - Optimality
  - Dependency (left)
  - Deviation (right)

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

$$f(\mathbf{x}) = \sum_{j=1}^{\ell} \beta_j \psi_j(\mathbf{x}|\mathbf{z}_j)$$

$$(\beta_j, \mathbf{z}_j) = \arg\min_{\beta, \mathbf{z}} \sum_{i=1}^{N} L(y_i, f_{j-1}(\mathbf{x}_i) + \beta \psi_j(\mathbf{x}_i|\mathbf{z}))$$

Prediction Model

$$y^k_{tot}(t_c) = \mathcal{P}\left(\{\mathbf{x}(t)\}^{t_c}_{t=t_c-m}\right), \text{ where } y^k_{tot}(t_c) := \sum_{t=t_c+1}^{t_c+k} y(t)$$

Results

- Slowdown from…
  - Other jobs
  - Endpoint congestion
- Forecasting Model
  - Low MAPE
  - Better as more runs are completed